An Automated Tool for the Design and Assessment of Space Systems

Lois M.L. Delcambre and Steve P. Landry
The Center for Advanced Computer Studies
University of Southwestern Louisiana
P.O. Box 44330
Lafayette LA 70504-4330
(318) 231-5697 (318) 231-6768
lmd@usl.usl.edu spl@usl.usl.edu

Abstract

Space systems can be characterized as both large and complex but they often rely on reusable subcomponents. One problem in the design of such systems is the representation and validation of the system, particularly at the higher levels of management. This paper describes an automated tool for the representation, refinement, and validation of such complex systems based on a formal design theory, the Theory of Plausible Design. In particular, this paper describes the steps necessary to automate the tool and make it a competent, usable assistant.

1. Introduction

The process of design relies heavily on human creativity and judgement. Design is particularly difficult when the artifact being designed is large and complex, such as with satellites, computer systems, etc. At the highest level, the design, development and evolution of such complex systems must be managed through appropriate validation and assessment techniques. Although numerous tools exist to aid in the detailed design and development of individual components through CAD technology, there is little automated support for management and development of complex systems at the higher levels.

The design and development of artifacts can be viewed as the process of satisfying a set of constraints or requirements. Constraints are satisfied either by detailed elaboration into subcontraints and/or by providing evidence that the constraint is satisfied. As an example, a new satellite program begins with the specification of the scientific aspects of the satellite, termed the *mission requirements*. The mission requirements must be refined so that the end-product will meet the scientific objectives. Thus, if the mission requirements demand a sensor with a particular sensitivity, the designer (or program manager) may choose an off-the-shelf component, with a track record of high sensitivity. The documented track record then provides *evidence* that the sensor sensitivity requirement has been met. However, there may be other constraints that interfere with the choice of the sensor. The selected sensor may require too much power, may be too large or heavy for the satellite, or may be so delicate that it will not survive the intended launch procedure. The fundamental problem in the development of complex systems is the representation, refinement, evolution, and assessment, and ultimately the validation of a myriad of diverse constraints.

This paper addresses this problem by describing an automated tool that relies on the Theory of Plausible Design, TPD [D89]. A plausible design, at any stage in the design process, is represented by a set of constraints with associated plausibility states organized into a directed, acyclic Constraint Dependency Graph (CDG). The

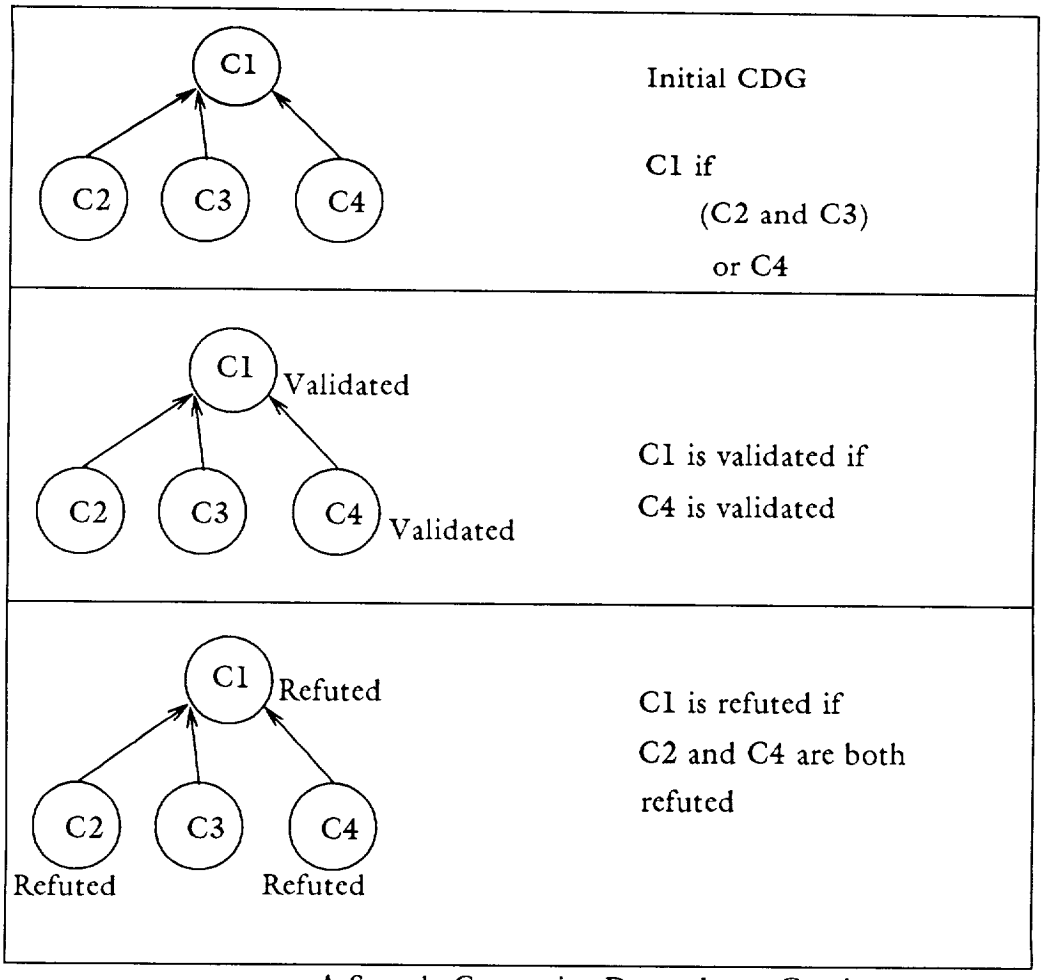
plausibility state takes the following values: unknown, assumed, validated, or refuted. The final two states, validated and refuted, are established through the presentation of evidence for or against the constraint, respectively. The CDG represents the successive refinement of constraints into subconstraints and supports the automatic computation of the plausibility state for each constraint.

The contribution of this paper is the description of how TPD can be used as the formal framework for an automated tool to manage, develop, refine, assess, and validate complex systems. TPD provides the framework for the tool through the CDG with automatic propagation of plausibility states. The remainder of this paper is organized as follows. TPD is presented in Section 2 along with several examples. The challenge is first to implement the CDG and plausbility state propagation and then to provide appropriate interfaces and structure to make the tool easy to use and to facilitate the development of correct systems. The major components of the automated tool are presented in Section 3. The paper concludes with a discussion of the work in progress, in Section 4.

2. The Theory of Plausible Design

The Theory of Plausible Design represents a design as a set of plausibility statements. Each plausibility statement contains a representation of the actual constraint, expressed in English or other, perhaps formal, language. The plausibility statement also includes the plausibility state for the constraint. When a constraint is first formulated, the state is initialized to ***unknown***. At this point, the designer may provide direct evidence that the constraint can be satisfied. On the other hand, the designer may develop one or more alternative refinements such that if the refinements can be satisfied then the original constraint will be satisfied. When a constraint is refined into one or more subconstraints, the plausibility state of the original constraint depends on the plausibility state of subconstraints. The details of how a given constraint is related to the alternatives described in its subconstraints is given by a well-formed formula connecting the subconstraints with *and* and *or*. Eventually, every constraint at the lowest level of detail (i.e. a constraint with no subconstraints) must be validated by the presentation of evidence. Evidence can be quite precise (e.g. a proof), heuristic (e.g. expert opinion), or experimental (e.g. results from simulation). One strength of TPD is the automatic propagation of the plausibility state from the lowest level upwards to all affected constraints. The propagation of plausibility state enforces the semantics associated with the well-formed formula of the non-leaf constraints. In addition to effectively capturing the current design state, TPD provides a paradigm that captures the design history and integrates various forms of evidence to validate constraints.

One important and equivalent view of the design is the representation of the plausibility statements (or constraints) with the Constraint Dependency Graph (CDG). In this view, the nodes of the graph represent the constraint, the plausibility state of the constraint, and the evidence that either refutes or validates the constraint. The arcs of the CDG represent the subconstraint relationship and includes a graphical representation of the *and* and *or* relationship. As an example, consider the CDG presented in Figure 2.1. The upward arcs that meet at a single point indicate that the subconstraints must all be validated in order to validate the constraint (i.e. the *and* relationship). Multiple sets of inward arcs indicate the *or* relationship.



A Sample Constraint Dependency Graph
Figure 2.1

At the lowest level of the CDG, individual constraints are validated or refuted through the presentation of evidence in favor or against the constraint, respectively. Once the plausibility state of constraints at the lowest level is either validated or refuted, then the state is propagated to the next higher level according to the appropriate connectives (***and*** or ***or***) implied by the CDG. For example in Figure 2.1, constraint C1 may be satisfied if C2 and C3 are satisfied or if C4 is satisfied. If evidence is presented for C4 to be validated, then the plausibility state propagation would then record C1 as validated. Similarly, if evidence were presented to refute both C2 and C4, then the plausibility state of C1 would also be refuted. These three scenarios are shown in Figure 2.1.

A significant problem faced by the program manager is that of assessment. Note that the process of design happens over time and that the assessment function relies primarily on the performance of human experts. The primary objective of the review team is to certify that the current design representation meets the original constraints. Among the major obstacles that the review team must overcome are the lack of direct connections from the current components of the design to the original constraints and

the information that describes the interactions among current designed components. TPD and the resulting design paradigm provides solutions to both of these obstacles by capturing a full history of the design and maintaining the interactions between components of the design at all phases.

3. The Automated Design and Assessment Tool

TPD provides a powerful framework to manage the design of complex systems. However, the automation of TPD to serve as a competent assistant (e.g. to the program manager) presents a number of challenges. The specific components of the tool that address these challenges are presented here. First, the components intended to support TPD directly are presented in Section 3.1. Then the opportunities for using application-specific structure in the form of a semantic network are presented in Section 3.2. Finally, the use of previous, plausible designs as a knowledge-base to guide and to facilitate the design process is discussed in Section 3.3.

3.1. Automating the Theory of Plausible Designs

TPD is intended to capture the design process and the design rationale through the constraints represented in the CDG. The TPD tool must interface (upward) to the designer and (downward) to other automated design/development systems. The user interface is a graphical interface with support for easy creation and manipulation of the CDG and a variety of abstraction mechanisms. The CDG is displayed graphically with the plausibility state at each node clearly highlighted. The design/refinement/validation process is driven by the current state of the CDG and thus the work remaining to complete the design is easily presented to the designer.

During the development of a satellite or other complex system, the program manager may develop the system specification (from the mission requirement) and then rely on other development groups (or subcontractors) to develop the satellite, the launch vehicle, and the ground system, for example. In this case, the current state of the CDG represents the specification of the entire system under design. The (downward) interface provides the appropriate constraints from the lowest level of the CDG as the formal specification for the detailed design and development of the subcomponent to be communicated to the subcontractors. Then, after the subcomponent is designed or implemented (e.g. in another automated system), standard practices of testing, simulation, etc. can be used to successfully validate that the completed design (or system) satisfies the associated leaf constraints. This evidence is then recorded in the CDG to complete the design.

The emerging design maintained by the TPD tool is valuable in as much as it accurately reflects the subtle interaction among constraints. Said another way, the benefits of TPD and the confidence in a validated plausible design require that all of the appropriate connections among constraints be explicitly recorded in the CDG. The tool includes a synonym facility that can make suggestions concerning potentially relevant constraints. As an example, the constraint concerning the sensitivity of the sensor ultimately affects the power system, the control system, and the size and weight of the satellite. These connections can be suggested by the tool based on the recognition of synonyms present in the statement of the constraint and subconstraints associated with the actual sensor selected. The final responsibility for a correct CDG ultimately rests with the designer. The synonym facility just tries to help.

3.2. Application-Specific Structure

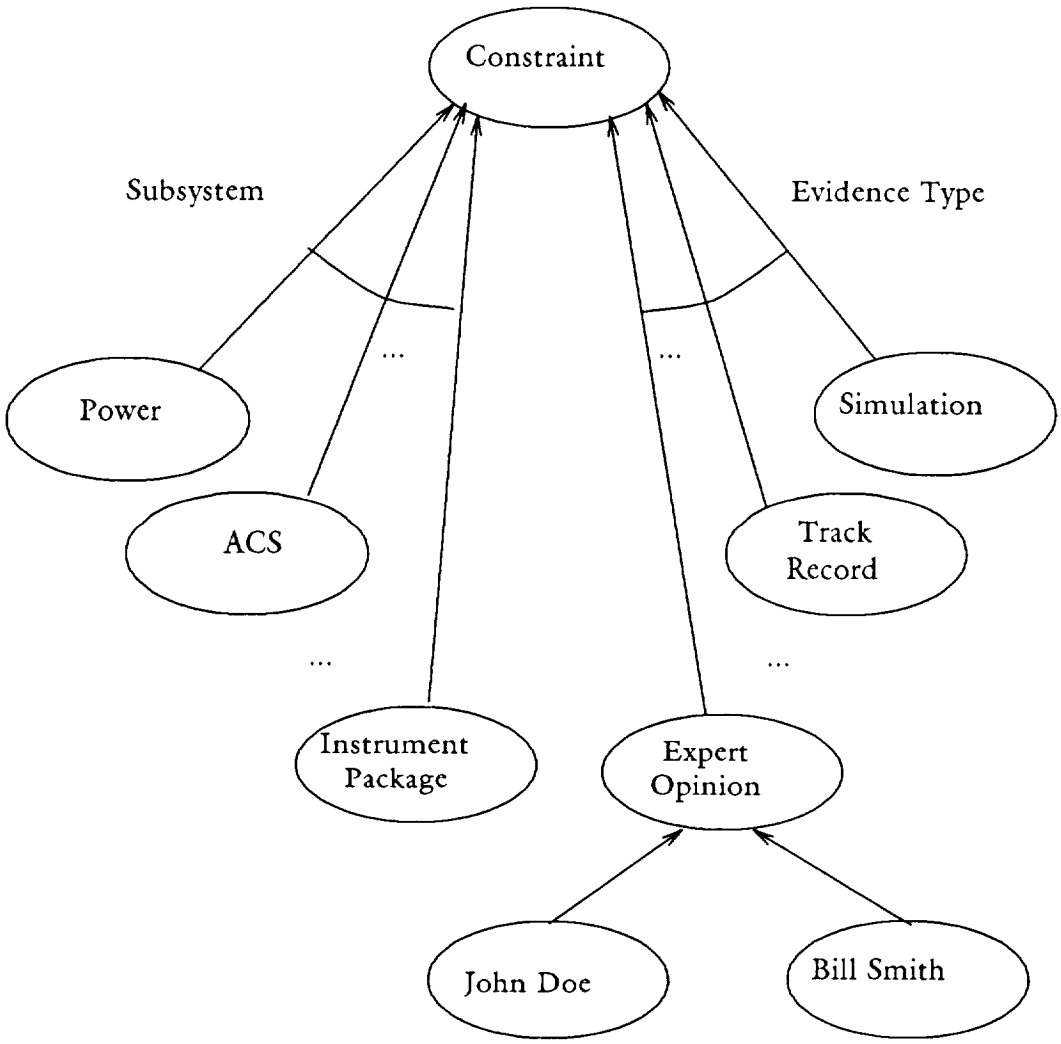
In general, a single CDG represents the system at any particular state in the design. The CDG may be arbitrarily deep, depending on the level of refinement of the various constraints. The CDG may also be arbitrarily large, depending on the complexity of the system. The CDG is built according to the refinement and interrelationship among constraints. As in the example concerning the sensor sensitivity, a single constraint may influence many other components of the system. Thus the CDG can include arbitrarily complex interconnections. The challenge then is to manage this complexity through the use of abstraction. The problem is: how can the tool help the designer focus his attention?

When the TPD tool is used for a particular application, additional structure can be used to facilitate the development process. The structure is introduced at a level above the CDG through the use of a semantic network. At the meta level, the nodes of the CDG can be grouped in a variety of ways. As an example, consider the semantic network in Figure 3.1. The individual constraints are all members of the most general class, Constraint (at the top of the semantic network). Additionally, the constraints can be placed in any or all of the subclasses. One way to group constraints is according to the subsystem that they belong to. For a satellite, the subsystem structure is shown on the left side of Figure 3.1 with three subclasses shown for Power, ACS (attitude control subsystem), and the Instrument Package. This set of subclasses is marked as being a "Subsystem" classification by the labeled arc. Similarly, the constraints can be classified according to evidence type, as shown on the right side of the figure. A given constraint could then be entered into the appropriate subsystem class and also into the appropriate evidence type class. If the evidence type is "expert opinion", then the constraint can also be entered into the subclass according to the actual expert. The structure shown in Figure 3.1 can be used to highlight the CDG according to the needs of the user. For example, if the user wants to work on the power subsystem, the user can request that only the constraints that are in the Power subclass in Figure 3.1 be shown. The remaining constraints in the CDG can then be made invisible. Similarly, the user may wish to highlight all constraints that were validated by Bill Smith. Then the constraints with an evidence type of "Expert Opinion" that were validated by "Bill Smith" could be highlighted (perhaps in color). The application structure can be subdivided (or classified) in many other ways. The strength of this approach is that the additional structure can be easily introduced at the meta level as shown in Figure 3.1 and then can be used to drive the user interface.

The structure of the CDG and also of the meta level suggests that a database style query language can be used to navigate around these structures. Within a CDG, a query language can be used to locate all subconstraints, all superconstraints, all assumed constraints, etc.

3.3. The Design History

If the TPD tool is used to develop multiple satellite systems, for example, then the design history serves as a valuable knowledge base. In particular, the previous designs can serve as a library for reusable subcomponents and also as a rich source of suggestions to solve design problems. For the first part, any subcomponent for which there exists a plausible design (e.g. a detailed design for the sensor), can be incorporated into the current design. This process is facilitated both by the application-



Application-Specific Meta Level Structure
Figure 3.1

specific structure and by the synonym facility. Each constraint in the CDG of the subcomponent can be labeled with the appropriate keywords (e.g. to indicate the appropriate subsystem of the satellite). Then, when the CDG for the subcomponent is merged with the CDG of the design in progress, the synonym facility can suggest the appropriate connections and can also populate the meta level classes.

Another powerful use of the history database is for access to lessons learned. A previous mission may have encountered a similar design problem and may have solved it. By the appropriate pattern matching process, the current CDG can be matched against portions of the CDG from previous designs. The evidence used to validate the previous design can then be suggested as a way to validate the current design. Note that the pattern matching process required to access the history is very complex, in general. However, within a specific domain with well-defined subsystems and components, a useful pattern-matching facility can be implemented. Once the appropriate portions of the history are located, then the CDG and the evidence used to validate or

refute previous design can be used to drive the current design and suggest possible testing procedures.

Finally, the tool is part of a knowledge-based framework that supports various design methodologies. The choice of what to do next can be viewed as the choice of which part of the CDG to concentrate on. The user may choose a top-down or bottom-up methodology or some combination. The user may be driven according to the constraints that currently have a "refuted" plausibility state. The criteria used to guide the design can be easily recorded in rules that act on the current state of the CDG. The use of the TPD tool to support design methodologies is described in more detail in [DL90].

4. Work in Progress

Plausibility theory has been formally defined and a number of case studies have been developed. However, so far all such plausible designs have been developed manually. The use of TPD as the basis for an automated tool is currently being investigated in conjunction with Orbital Systems, Ltd. for an Air Force project [O90]. The strength of TPD stems from the automatic maintenance of plausibility states. When the CDG is correctly represented, then the presentation of evidence to validate (or refute) subconstraints is automatically reflected in the entire CDG. *All* affected constraints are immediately identified. This paper describes the steps required to make the tool usable. The design and development of the tool is currently underway. The work with the Air Force includes the collaboration of a variety of space system customers through the clientele of Orbital Systems, Ltd.

References

[D89] Dasgupta, Subrata, "The Structure of Design Processes", Advances in Computers, Vol. 28, 1989.

[DL90] Delcambre, L.M.L., Landry, S.P., and Dasgupta S., "A Knowledge-Based Framework for an Integrated Engineering Design System", 1990 Eastern Multi-Conference, April 1990.

[O89] Orbital Systems, Ltd., "Reduction of Space Costs Using Expert System Technology", SBIR Phase II Grant Proposal, Air Force, Mar. 1989.